DOCUMENT RESUME

ED 110 003 IR 002 271

AUTHOR Hedayah, Mohamed M.
TITLE An Introduction to Decision Logic Tables.
INSTITUTION University of Southern California, Los Angeles. School of Library Science.
PUB DATE 74
NOTE 25p.

EDRS PRICE MF-$0.76 HC-$1.58 PLUS POSTAGE
DESCRIPTORS *Branching; Computer Programs; *Decision Making; Expectancy Tables; *Library Acquisition; Library Science; Logic; *Logical Thinking; Programing
IDENTIFIERS *Decision Tables

ABSTRACT

The use of decision tables--which are a means of linking decision rules for actions to specific sets of prior conditions--in information systems design and development is described. Procedures for preparing decision tables are presented together with examples of their application in the context of library acquisitions. A bibliography is also provided. (DGC)

AN INTRODUCTION TO DECISION LOGIC TABLES

A Research Paper Presented to
Dr. Edward Kazlauskas

SCHOOL OF LIBRARY SCIENCE
UNIVERSITY OF SOUTHERN CALIFORNIA

In partial fulfillment
of the requirement for the course

L.S. 563
"Method and Technology in
Information Science"

by

Mohamed M. Hedayah

Fall 1974

PREFACE

This paper makes no pretense of being a definitive work. Except for the section dealing with types of tables, the rules and examples given are all for decision tables in limited entry format, the type most commonly used.

No attempt has been made to explain advanced theories; however, for those who wish to explore them independently, a list of "Works Cited" is provided.

Any attempt such as writing this paper is always more successful when the author receives assistance from others more knowledgeable in the field. That is why I wish to express my deep gratitude to Dr. Edward Kazlauskas for his invaluable advices.

I wish to thank also the staff members in the whole library system of the University of Southern California, the Los Angeles City, and the Los Angeles County where I found help when I needed it during my research.

Mohamed M. Heddyah

Los Angeles
December, 1974

CONTENTS

I. INTRODUCTION

1. Need

The need that exists today for a standard method for defining problems has lead to experimentation with decision tables. To date no approach seems to offer as much success particularly in those systems that usually have a number of interrelated conditions that require specific actions for which flow charts and narratives are cumbersome.

Computer programming demands a precise, written definition of all instructions, from routine steps to complex logic decisions, all of which are based on a myriad of facts and conditions. By using the Decision Logic Tables technique, these facts and conditions can be stated in a tabular form in the logical sequence, and each condition can be tested before a certain action is taken. This technique serves multiple purposes:

- it clearly defines the sequence of the decisionmaking process
- it eliminates repetitive narrative description of programming steps that are necessary for varying conditions
- it simplifies the means of communication between the systems analyst and the programmer by reducing the amount of narrative information
- it makes clear that which is difficult to define and hard to interpret
- and it also serves as a valuable aid in the documentation phase.

2. History

The origin of decision tables springs partly from the general use of tables to present information effectively, and partly from the development of truth tables to define logic.

However, the history of their use in data processing and the computer languages adapted for them can be summarized as follows:

First for some definitions:

- A Decision Table Processor is a program for translating decision tables into executable computer programs
- An Interpretive Routine is a single computer program capable of analyzing any decision table and then selecting an appropriate subroutine to process actual data
- A Compiler analyzes statements in a source language, generates machine language instructions, and compiles these instructions into an operational program

• A Translator or Pre-Processor translates decision tables into an existing source language which can be processed by compilers already available.

1. In 1957, General Electric initiated a study which culminated in the development of " decision tables " and a computerized method for solving them. An improved processor and language called TABSOL (TABular Systems Oriented Language) which is an example of a decision table compiler, was implemented on the GE 225 in early 1961.
2. In 1958, Sutherland Company developed its tables strictly as an aid to systems analysis and documentation leaving the solution of the table to the programmer.
3. In 1959, Hunt Foods and Industries began using decision tables as an aid in man-to-man communication.
4. In 1960, the CODASYL (COnference on DAta SYstems Languages) Systems Group after reviewing several approaches to the objective of developing a machine-independent, systems-oriented language, began to study decision tables ——
5. In 1962, the CODASYL Systems Group study resulted in a decision table language known as DETAB-X (DEcision TABles, EXperimental).
6. In 1965, the SIGPLAN (Special Interest Group for Programming LANguages) of the Los Angeles Chapter of the Association for Computing Machinery (ACM) appointed a working group who developed DETAB-65.
7. Other work on decision tables was independently taken up by such companies as:
   - North American Aviation 'DETAB-66'
   - Dow Chemical Company 'DETAB-67'
   - The Insurance Company of North America 'LOBOC' (LOgical Business Oriented Coding).
   - Rand Corporation 'FORTAB' (it couples decision tables with Fortran)
   - Bell Telephone of Canada 'PET' (Pre-processor for Encoded Tables)
   - R.L.Martino and Co.,Inc. 'DETRAN' (DEcision TRANslator).

Both PET and DETRAN are examples of processors which effectively extend decision table capabilities to the computer level.

II. BASIC STRUCTURE

A decision table is a formal method for recording the conditions applying to a particular situation and the actions which ensue from their different possible combinations. In other words, the decision table shows all the appropriate decision rules governing the situation at hand where each decision rule represents the relationship between a set of conditions and their associated actions and is of the form:
IF A and B and C, THEN X and Y and Z
(CONDITIONS) (ACTIONS)

While the format of decision tables can be of many types,[1] Fig.II.1 is an illustrated decision table format that is in general use.

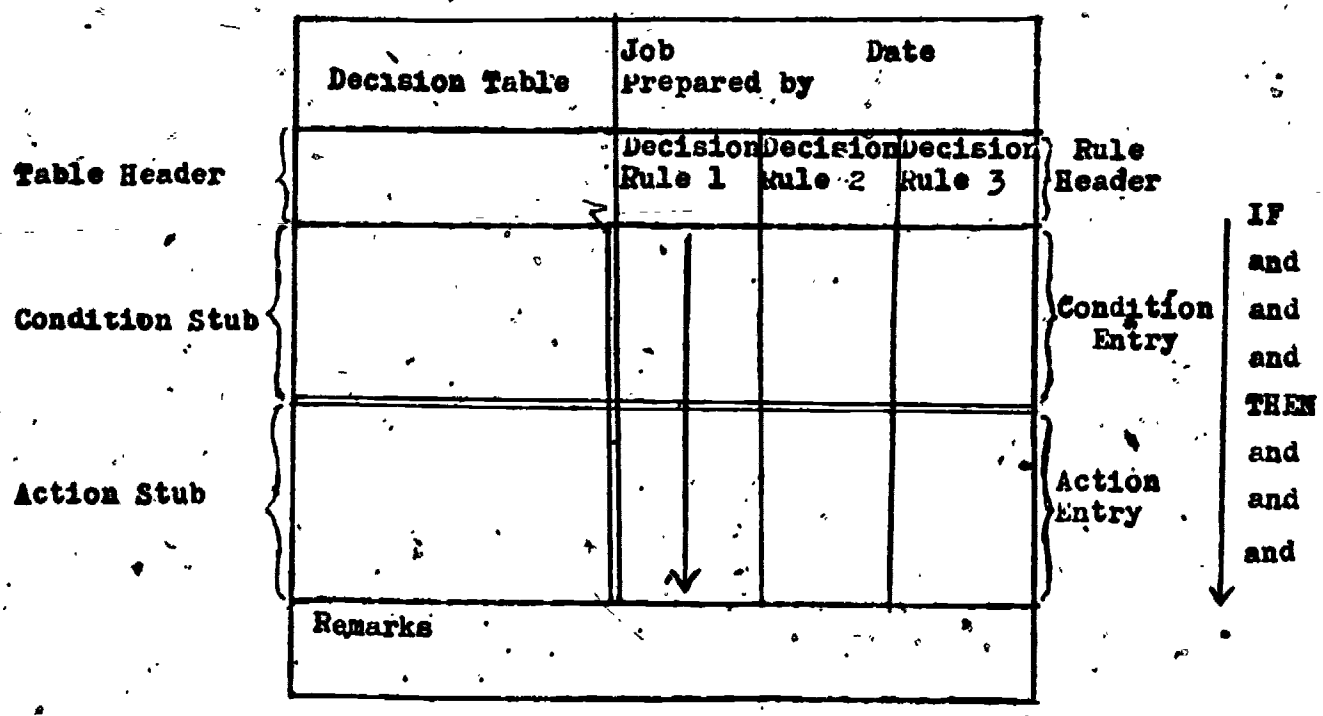

Fig.II.1
An Illustrated
Complete Decision Table Format

There are three main parts to decision tables: the table header or identification portion, the body and the rule header.

1. The Table Header contains the descriptive title of the system or

---

[1] Martin L. Rubin, ed., Handbook of Data Processing Management, 6 vols. (Princeton: Brandon/Systems Press, 1970-), vol. 3: Systems Life Cycle Standards: Forms Method, by P. Zuckerman, pp. 91-98.

procedure covered by the table. It is the upper left portion of the table.

2. The Body it consists of four basic sections. A double horizontal line divides the decision table into two major sections: Conditions and Actions as shown in Fig.II.2. A double vertical line divides the table into two other major sections: Stub to the left and Entry to the right as shown in Fig.II.3. These double lines are an invaluable aid in reading the table. The stub portion describes and names the conditions and actions in which we are intersted; the entry portion specifies the logical relationships. Horizontal levels called rows, run across the entire table. To aid in identifying them, they may be assigned letters or numbers. The entry portion is subdivided by vertical lines to form columns called rules. Fig.II.4 shows the four basic sections:

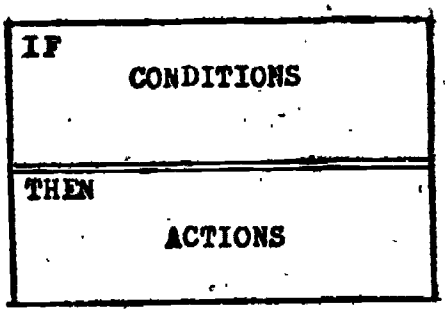

Fig.II.2 Conditions and Actions

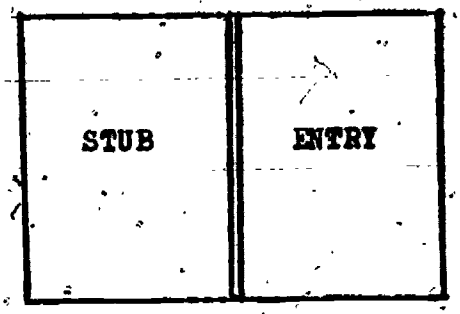

Fig.II.3 The Stub and the Entry

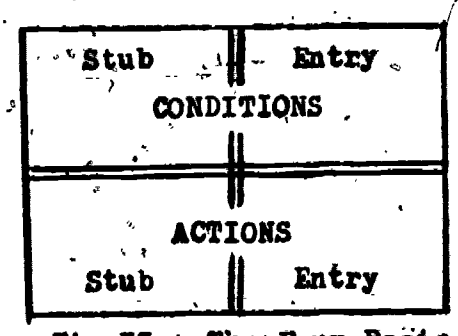

Fig.II.4 The Four Basic Sections of a Decision Table Body

i. Condition Stub it is the upper left quadrant. Here named variables being examined are listed in a question form.

ii. Condition Entry it is the upper right quadrant. Here particular values for those conditions variables are specified on the corresponding lines (rows). For a limited entry table responses are restricted to "Y" to indicate yes, "N" to indicate no. A condition entry is left blank only if the condition does not apply or if in the presence of other conditions a yes or no cannot affect our actions.

iii. Action Stub(the lower left quadrant.)It is here that possible actions are described.

iv. Action Entry(the lower right quadrant.) It is here that the executions of the actions are specified. For a limited entry table, the only permissible entry here is an "X" , on the corresponding line next to a listed action to indicate "Take this action". A blank next to an action may be interpreted as "Do not take the action shown".

The stubs may contain a number or different conditions and actions as required. A particular instance (combination) of specified conditions and their associated actions constitute a decision rule. Essentially, a decision rule is an "If ... Then ..." statement. Decision rules are set in vertical columns in the entry portion of the table. Each decision rule can be read in the direction of the arrows shown in Fig.II.1, and is given a number for identification purposes. To interpret the table, one single rule has to be examined at a time, together with the information on the stub; only one rule can be satisfied in a single pass (examination) through the table.

3. The Rule Header it is above the entry portion of the table and it is here that numbers are given to the different decision rules.

The last row or rows of the table are used for any necessary remarks.

Types of Tables

The decision tables are classified according to the approach taken in the entries into three types:

1. Limited Entry Tables Here condition entries are limited to "Y"s , "N"s, or blanks, and action entries are limited to "X"s or blanks. Although limited entry tables tend to be longer than the other two types, they are built on binary logic patterns, which are especially well suited for computer applications.
2. Extended Entry Tables Here the statements in the stub portion of the table are extended into the entry portion. The stub portion identifies the variables whose values are assigned in the entry portion. This type lends itself to problems where a few variables can have many values.
3. Mixed Entry Tables When limited entry form and extended entry form are combined into a single table, the resulting table is said to be in mixed-entry form. Even though these two forms may be combined, one form only must be used exclusively within each horizontal row of the table.

An extended or a mixed entry table may always be changed into a limited entry table by listing each of the values of a variable as a separate condition or action and assigning "Y"s, "N"s, or "X"s to the appropriate rules. This format is more precise, since each variable can have only one of two values.

III. AN EXAMPLE FROM THE LIBRARY

Before we give our example for a library procedure, let us mention some of the characteristics of a procedure description:

- Possible conditions isolated
- Possible actions isolated
- The combinations of conditions calling for different actions are described
- The actions called for are described
- If a combination of conditions calls for a series of actions, the action sequence is specified.

The effectiveness of a procedure description is increased if:

- Language is standardized
- Duplication is eliminated
- All possibilities are covered
- Redundancy is eliminated
- Contradiction is eliminated
- The document is readable (understandable)

Decision tables are a method for presenting procedure descriptions in a way such that the attainment of the above goals is maximized. For our example we shall start with a narrative as a procedure description and try to construct, step by step, the appropriate decision table for it putting in mind the attainment of these goals all the time. Some of the steps will be discussed in this section and the rest will be detailed in the next one.

Here now is the narrative for our library procedure example:

Acquisition Searching for Book Duplicates

When a book request is received in the preorder search subsystem, it is first searched in the main catalog to see if it is in the collection. If it is found in the collection and the request is not for an added copy, it is recorded as a duplicate and is returned to the requestor. If it is in the collection and the request is for an added copy, it is searched in the In-Process file, if it is found in this file, it is recorded as a duplicate and returned to the requestor. If it is not found in the In-Process file, it is recorded as an added copy, and the verified bibliographic data are added on the request slip and the search is continued.

If the book is not found in the collection, and the request is not for an added copy, and it is found in the In-Process file, it is recorded as a duplicate, and is returned to the requestor. If the book is not in the

collection and the request is for an added copy and is found in the In-Process file, it is recorded as an added copy and the verified bibliographic data are added on the request slip and the search is continued.

If the book is not in the library and is not in the In-Process file, the title is searched in the LC proof-card file. If the LC card is not located, search is continued. If the LC card is located, the main entry on the request slip is checked against the proof card; if it is the same, search is continued. If it is different, the main entry on the request slip is corrected to conform with the proof card, and the main catalog and the In-Process file are rechecked.

Before we start to isolate the conditions and actions, if we observe the last paragraph in our narrative we can see that for a unique combination of conditions (the book is neither in the collection nor in the In-Process file), a totally new procedure segment (searching the LC proof-card file), is called for. Furthermore, for a particular combination of conditions in this new procedure segment (the main entry on the request slip does not conform with the LC proof card), we have to correct it and then start allover again to see if the book is in the collection etc.

Fortunately, the decision table technique offers great flexibility through the "Go to Table X" feature. This feature not only helps prevent cramming of actions into one table, but also simplifies subsequent changes that may be necessitated by redesign. One or the actions on one table might refer to another table with another set of conditions and actions. With the exception of the return of control from a closed table (the discussion of which is beyond the scope of this paper), all tables are entered from the top.

By using this valuable feature, we will be able to link two decision tables together in order to describe effectively the procedure in our example.

Reading through our narrative, we can:

1. Isolate the conditions
2. Isolate the actions

for each of the two tables.

Following is the same narrative with the conditions underlined by single lines and the actions by double lines.

P.S. Notice that the actions that might in fact be replaced later by the "Go to Table X" statements (simply referring one table to the other), are underlined by broken double lines.

When a book request is received in the preorder search subsystem, it is first searched in the main catalog to see if it is in the collection. If it is found in the collection and the request is not for an added copy, it is recorded as a duplicate and is returned to the requestor. If it is in the collection and the request is for an added copy, it is searched in the In-Process file, if it is found in this file, it is recorded as a duplicate and returned to the requestor. If it is not found in the In-Process file, it is recorded as an added copy, and the verified bibliographic data are added on the request slip and the search is continued.

If the book is not found in the collection, and the request is not for an added copy, and it is found in the In-Process file, it is recorded as a duplicate, and is returned to the requestor. If the book is not in the collection and the request is for an added copy and is found in the In-Process file, it is recorded as an added copy and the verified bibliographic data are added on the request slip and the search is continued.

If the book is not in the library and is not in the In-Process file, the title is searched in the LC proof-card file. If the LC card is not located, search is continued. If the LC card is located, the main entry on the request slip is checked against the proof card; if it is the same, search is continued. If it is different, the main entry on the request slip is corrected to conform with the proof card, and the main catalog and the In-Process file are rechecked.

Now we can lift the isolated conditions and actions from the narrative description and put them into two separate lists as shown in the next page.

P.S. Notice that the conditions and actions for each of our prospective tables, are separated by broken double lines and are given sequent numbers.

Conditions	Actions
1. It is found in the collection	1. Is recorded as a duplicate
2. Request is not for an added copy	2. Is returned to the requestor
3. In the collection	3. Is recorded as a duplicate
4. Request is for an added copy	4. Is returned to the requestor
5. It is found in this file (the In-Process file)	5. Is recorded as an added copy
6. Not found in the In-Process file	6. Verified bibliographic data are added on the request slip
7. Book is not found in the collection	7. Search is continued
8. Request is not for an added copy	8. Is recorded as a duplicate
9. It is found in the In-Process file	9. Is returned to the requestor
10. The book is not in the collection	10. Is recorded as an added copy
11. Request is for an added copy	11. Verified bibliographic data are added on the request slip
12. Is found in the In-Process file	12. Search is continued
13. Book is not in the library	13. Title is searched in the LC proof-card file
14. Is not in the In-Process file	
==========	==========
1. LC card is not located	1. Search is continued
2. LC card is located	2. Search is continued
3. Main entry on the request slip is the same (as the proof card)	3. Main entry on the request slip is corrected
4. It is different	4. Main catalog and the In-Process file are rechecked

Now as we have isolated the conditions and actions, we should:

3. Standardize the language: One thing that can be observed about the above conditions and actions is that some of them talk about the same thing in different ways. For example conditions 10 and 13 for the first table above are the same. Similarly, we can now replace action 13 for the first table by the "Go to Table 2" statement, and action 4 for the second table by the "Go to Table 1" statement.

Here are the conditions and actions, for each table, with the language standardized and the conditions written in question form.

Table 1

Conditions

1. [illegible] in collection?
2. [illegible]est not for added copy?
3. Book in collection?
4. Request for added copy?
5. Book in In-Process file?
6. Book not in In-Process file?
7. Book not in collection?
8. Request not for added copy?
9. Book in In-Process file?
10. Book not in collection?
11. Request for added copy?
12. Book in In-Process file?
13. Book not in collection?
14. Book not in In-Process file?

Actions

1. Record as duplicate
2. Return to requestor
3. Record as duplicate
4. Return to requestor
5. Record as added copy
6. Add verified bibliographic data on request slip
7. Continue search
8. Record as duplicate
9. Return to requestor
10. Record as added copy
11. Add verified bibliographic data on request slip
12. Continue search
13. Go to Table 2

Table 2

Conditions

1. LC card not located?
2. LC card located?
3. Same main entry?
4. Different main entry?

Actions

1. Continue search
2. Continue search
3. Correct main entry on request slip
4. Go to Table 1

Now as we have standardized the language, we are in a position to:

4. Eliminate duplication
5. Eliminate negative conditions

It is easy to see that many of the conditions and actions are duplicates. For example conditions 1 and 3 for table 1 are duplicates. Besides, as with respect to each condition, we are going to indicate whether or not it is present; that is, for each condition we are going to answer YES or NO, we should, to avoid duplication, eliminate negative conditions. For example, conditions 1 & 2 and conditions 3 & 4 for table 2, are negative to each others. Here are the conditions and actions, for each table, after eliminating the duplicates and the negative conditions.

Table 1

Conditions

1. Book in collection?
2. Request for added copy?
3. Book in In-Process file?

Actions

1. Record as duplicate
2. Record as added copy
3. Add verified bibliographic data on request slip
4. Return to requestor
5. Continue search
6. Go to Table 2

Table 2

Conditions

1. LC card located?
2. Same main entry?

Actions

1. Continue search
2. Correct main entry on request slip
3. Go to Table 1

Here are the two linked decision tables for our example:

Acquisition Searching for Book Duplicates. Table 1 (of 2)	1	2	3	4	5	6
. Book in collection?	Y	Y	Y	N	N	N
. Request for added copy?	Y	Y	N	Y	N	
. Book in In-Process file?	Y	N		Y	Y	N
. Record as duplicate	X		X		X	
. Record as added copy		X		X		
. Add verified bibliographic data on request slip		X		X		
. Return to requestor	X		X		X	
. Continue search		X		X		
. Go to Table 2						X

Acquisition Searching for Book Duplicates. Table 2 (of 2)	1	2	3
. LC card located?	Y	Y	N
. Same main entry?	Y	N	
. Continue search	X		X
. Correct main entry on request slip		X	
. Go to Table 1		X	

Let us now review some of the points we have studied so far about decision tables and how they are demonstrated in our example.

. The conditions are listed in question form in the condition stub.

. The actions are listed in the action stub.

. For limited entry tables the entries are limited to "Y"s, "N"s, and blanks in the condition entry, and "X"s and blanks in the action entry.

. The table header holds the title of the system or procedure together with the respective identification number of the table if more than one table have to be linked together.

. The rule header holds the identifying numbers of the different decision rules.

. The decision rules are set in vertical columns in the entry portion of the table.

. In table 1, the first rule (column 1) says: if the book is in the collection, and the request is for an added copy, and the book is found in the In-Process file, then record as a duplicate, and return to the requestor.

. In table 1, the last rule (rule 6) says: if the book is neither in the collection nor in the In-Process file, Go to Table 2.

. In table 2, the second rule says: if the LC card is located, and if the main entry on the request slip does not conform with the proof card, correct main entry on the request slip, and Go to Table 1.

. As we have said before, with only one exception, all tables are entered from the top. That is, if we follow up the path of the second rule in table 2, it will always be examined against the rules in table 1 from left to right, rule one through six, exactly in that particular order. If rule 6 in table 1 is met again, and we follow it through table 2,

it will first be examined against rule one, which it will always meet. This is because the LC card has been located earlier, and the main entry has been corrected to conform with the LC proof card, which satisfies this rule, and search is continued.

IV. BASIC STEPS IN CONSTRUCTING DECISION LOGIC TABLES

While the basic steps can be put in a precise, compact and comprehensive manner[1], in this section we shall discuss and give examples of the detailed necessary steps for constructing a perfect decision table. Notice that the first five steps we have already discussed and applied in the previous section.

1. Isolate the conditions.
2. Isolate the actions.
3. Standardize the language.
4. Eliminate duplication.
5. Eliminate negative conditions.
6. List Actions in execution order: The logic of the table is not dependent on the order in which the conditions rows are written or the order in which the rules are formulated. Actions, however, are performed in the sequence in which they occur. In many procedures, actions must be taken in a specific sequence. In decision tables, this sequence is generally indicated by the sequence the actions are listed in the action stub. In those instances where the actions sequence for one rule is different than the actions sequence for another, the sequence is indicated by substituting numbers for the "X"s and numbering the actions in sequence. Our example does not need this refinement, but is shown with it in Fig.IV.1 to illustrate the point.

Table 2 (of 2)	1	2	3
.LC card located?	Y	Y	N
.Same main entry?	Y	N	
.Continue search	1		1
.Correct main entry on request slip		1	
.Go to table 1		2	

Fig.IV.1

[1] Robert M. Hayes and Joseph Becker, Handbook of Data Processing for Libraries (New York: Becker and Hayes, 1970), pp. 158-160.

7. Check for Completeness: A table is complete in a mathematical sense only if all mathematically possible combinations are covered. We can use a simple formula to determine the number of possible combinations any table must cover: where only two variables are present, if n equals the number of conditions and r equals the number of rules, then $r=2^n$.

However, impossible or irrelevant conditions should be combined with possible or relevant ones, through the use of blanks. At least one rule, however, should not contain blanks. A blank in any rule is a sign that compression has taken place.

Definition

- A Pure Rule is a rule where all the condition entries are either a Y or a N.
- A Mixed Rule is a rule where any of the condition entries is a blank.

To calculate the number of combinations contained in any mixed rule, we can use another similar formula. Assuming two variables, where b is the number of blanks in the condition entry of a given rule and c is the number of combinations, then $c=2^b$.

By calculating the number of combinations encompassed by each rule, we can arrive at a rule count to verify completeness. Fig.IV.2 illustrates Table 1 expanded to allow for all the mathematically possible combinations.

Table 1(of 2)	1	2	3	4	5	6	7	8
.Book in collection?	Y	Y	Y	Y	N	N	N	N
.Request for added copy?	Y	Y	N	N	Y	Y	N	N
.Book in In-Process file?	Y	N	Y	N	Y	N	Y	N
.Record as duplicate	X		X	X			X	
.Record as added copy		X			X			
.Add verified bibliographic data on request slip		X			X			
.Return to requestor	X		X	X			X	
.Continue search		X			X			
.Go to table 2						X		X

Fig.IV.2

8. Eliminate Redundancy and Contradiction: One of the advantages of decision tables over other forms of procedure description is the ability to apply a test to detect redundancy and contradiction. Besides, several laws can be applied to eliminate redundancy. (Contradiction is something we have to eliminate ourselves, since it indicates confusion with respect to the procedure being described.)

All the laws for eliminating redundancy have to do with rules having the same actions.

First Law: .IF WITH THE EXCEPTION OF ONE CONDITION, TWO SUCH RULES (rules having the same actions) HAVE THE SAME CONDITION ENTRIES,

. AND IF FOR THAT ONE CONDITION (the one not having the same entries) ONE RULE HAS A YES ENTRY AND THE OTHER A NO ENTRY,

. THEN THE RULES CAN BE COMBINED INTO ONE WITH THE ENTRY FOR THAT CODITION (the one having a yes entry in one rule and a no entry in the other) BECOMING INDIFFERENT (blank).

Now if we notice the rules in table 1 (Fig.IV.2), we can see that rules 3 & 4, and rules 6 & 8 are redundant according to the first law. Fig.IV.3 shows the table with these rules combined as specified in the law.

Table 1(of 2)	1	2	3	4	5	6
.Book in collection?	Y	Y	Y	N	N	N
.Request for added copy?	Y	Y	N	Y	N	
.Book in In-Process file?	Y	N		Y	Y	N
.Record as duplicate	X		X		X	
.Record as added copy		X		X		
.Add verified bibliographic data on request slip		X		X		
.Return to requestor	X		X		X	
.Continue search		X		X		
.Go to table 2						X
			3&4			6&8

Fig.IV.3

Now that we have applied the first law, we are in a position to state the test for detecting redundancy and contradiction:

EACH PAIR OF RULES REMAINING (after application of the first law) MUST HAVE AT LEAST ONE CONDITION FOR WHICH ONE RULE HAS A YES ENTRY AND THE OTHER A NO ENTRY.

If a pair of rules meets this test, they are said to be <u>independent</u> of each other. If there are any rule pairs that are <u>dependent</u> (not independent), then the decision table still contains redundancy and/or contradiction.

. A dependent rule pair with the same actions indicates redundancy.

. A dependent rule pair with different actions indicates contradiction.

An inspection of Table 1 in Fig.IV.3 indicates that all the rules are independent of each other i.e. it is free of redundancy and/or contradiction.

If the test indicates the presence of some dependent pair rules with the same actions, the application of further laws for reducing redundancy becomes appropriate.

Second Law: IF ONE RULE IS PURE AND THE OTHER IS MIXED, THE PURE RULE IS CONTAINED IN THE MIXED RULE.

In the illustrated example in Fig.IV.4, this law is applicable to mixed rule 6 and pure rule 7 which are redundant. Fig.IV.5 shows the table after removal of rule 7.

	1	2	3	4	5	6	7
.Condition 1	Y	Y	Y	N	N	N	N
.Condition 2	Y	N		Y	N		Y
.Condition 3	Y		N	Y		N	N
.Action 1		X	X		X	X	X
.Action 2	X	X	X				
.Action 3				X	X	X	X

Fig.IV.4

	1	2	3	4	5	6
.Condition 1	Y	Y	Y	N	N	N
.Condition 2	Y	N		Y	N	
.Condition 3	Y		N	Y		N
.Action 1		X	X		X	X
.Action 2	X	X	X			
.Action 3				X	X	X

Fig.IV.5

Third Law: IF BOTH RULES ARE MIXED, THERE IS AT LEAST ONE PURE RULE, COMMON TO BOTH THAT CAN BE ELIMINATED FROM ONE OF THE ORIGINAL RULES.

In Fig.IV.5 this law is applicable to rules 2 & 3 and rules 5 & 6.

Fig.IV.6 shows rules 2 & 3 expanded into the pure rules that made them up where we can see the pure rule that they have in common.

	1	A 2	B	A 3	B	4	5	6
.Condition 1	Y	Y	Y	Y	Y	N	N	N
.Condition 2	Y	N	N	N	Y	Y	N	
.Condition 3	Y	Y	N	N	N	Y		N
.Action 1		X	X	X	X		X	X
.Action 2	X	X	X	X	X			
.Action 3						X	X	X

Fig.IV.6

	1	2	3B	4	5	6
.Condition 1	Y	Y	Y	N	N	N
.Condition 2	Y	N	Y	Y	N	
.Condition 3	Y		N	Y		N
.Action 1		X	X		X	X
.Action 2	X	X	X			
.Action 3				X	X	X

Fig.IV.7

In Fig.IV.7 we eliminated one of the two common rules - rule 3A, and collapsed rules 2A and 2B back down into our old rule 2.

The same procedure is made again with rules 5 & 6 in Fig.IV.7 and is shown in Fig.IV.8 and Fig.IV.9.

	1	2	3	4	5 A	5 B	6 A	6 B
.Condition1	Y	Y	Y	N	N	N	N	N
.Condition2	Y	N	Y	Y	N	N	N	Y
.Condition3	Y		N	Y	Y	N	N	N
.Action 1		X	X		X	X	X	X
.Action 2	X	X	X					
.Action 3				X	X	X	X	X

Fig.IV.8

	1	2	3	4	5	6 B
.Condition1	Y	Y	Y	N	N	N
.Condition2	Y	N	Y	Y	N	Y
.Condition3	Y		N	Y		N
.Action 1		X	X		X	X
.Action 2	X	X	X			
.Action 3				X	X	X

Fig.IV.9

We now have a decision table in Fig.IV.9 in which all the rules are independent i.e. there is no contradiction and/or redundancy between rule pairs.

9. Include the Else Rule: One way to assure completeness is to incorporate the else rule. An else rule says: if none of the specified rules hold, then follow a certain procedure. There are three main features to the else rule: i. it is always the last rule

ii. it has no condition entries

iii. it must specify some action.

While the else rule may provide a convenient short cut in the early stages of problem analysis, there is always the danger that it can become a "catch-all." Since the purpose of using decision tables is to provide a clear-cut definition of what is to be done in all situations, the else rule should be avoided wherever possible.

Table 1(of 2)	1	2	3	4	5	6	E
.Book in collection?	Y	Y	Y	N	N	N	L
.Request for added copy?	Y	Y	N	Y	N		S
.Book in In-Process file?	Y	N		Y	Y	N	E
.Record as duplicate	X		X		X		
.Record as added copy		X		X			
.Add verified bibliographic data on request slip		X		X			
.Return to requestor	X		X		X		
.Continue search		X		X			
.Investigate error							X
.Go to table 2						X	

Fig.IV.10

Fig.IV.10 shows Table 1 of our example with the else rule incorporated. Incidentally, it should be pointed out that in our particular example, the else rule is redundant, because all the possibilities have been covered.

10. Optimize Searching: With all the pieces for each table assembled, our next step is to arrange them in the best possible order to minimize the "searching time" to locate the applicable rule. Our approach is to:

i. Arrange the condition rows so that the row with the fewest blanks appears first. Fig.IV.11 shows the condition half of Table 1 (unsorted). Fig.IV.12 shows it with the condition rows sorted.

	1	2	3	4	5	6
.Book in In-Process file?		N	Y	Y	N	Y
.Request for added copy?	N		Y	N	Y	Y
.Book in collection?	Y	N	N	N	Y	Y

Fig.IV.11

	1	2	3	4	5	6
.Book in collection?	Y	N	N	N	Y	Y
.Request for added copy?	N		Y	N	Y	Y
.Book in In-Process file?		N	Y	Y	N	Y

Fig.IV.12

ii. Once the rows are sorted, the next step is to sort the rules. For the purposes of the sort, we assign a value for each entry in the condition half based on Y>N>blank. We will then arrange the columns so that the ones with the greatest values come first; as a result the "Y"s will be sorted at the upper left portion of the condition entry. (We could just as easily take the negative approach). Working down the table row by row, the rules should be sorted on this basis: Y > N > blank. Examining the first row, Fig.IV.13 shows our example (Fig.IV.12) sorted. Examining the second row, Fig.IV.14 shows our example sorted. Examining the third row, Fig.IV.14 shows our example completely sorted. Fig.IV.15 shows the full table completely sorted with the rules assigned sequent numbers.

	1	5	6	2	3	4
.Book in collection?	Y	Y	Y	N	N	N
.Request for added copy?	N	Y	Y		Y	N
.Book in In-Process file?		N	Y	N	Y	Y

Fig.IV.13

	6	5	1	3	4	2
.Book in collection?	Y	Y	Y	N	N	N
.Request for added copy?	Y	Y	N	Y	N	
.Book in In-Process file?	Y	N		Y	Y	N

Fig.IV.14

Acquisition Searching for Book Duplicates, Table 1(of 2)	1	2	3	4	5	6
.Book in collection?	Y	Y	Y	N	N	N
.Request for added copy?	Y	Y	N	Y	N	
.Book in In-Process file?	Y	N		Y	Y	N
.Record as duplicate	X		X		X	
.Record as added copy		X		X		
.Add verified bibliographic data on request slip		X		X		
.Return to requestor	X		X		X	
.Continue search		X		X		
.Go to table 2						X

Fig.IV.15

Decision tables lend themselves to optimization for computer efficiency. Optimization consists of:

- minimizing the number of branching instructions in memory;
- minimizing the average number of branching instructions which will be executed.

V. WORKS CITED

1. Association for Computing Machinery. Decision Tables for Computer System Design and Programming. New York: ACM, 1968.
2. _____. Decision Tables for Computer System Design and Programming: Readings and Bibliography. New York: ACM, 1967.
3. _____. Decision Tables for Computer System Design and Programming: Course Outline. New York: ACM, 1967.
4. Gildersleeve, Thomas Robert. Decision Tables and their Practical Application in Data Processing. Englewood Cliffs: Prentice Hall, 1970.
5. Hughes, Marion L.; Shank, Richard M.; and Stein, Elinor Svendsen. Decision Tables. Wayne: MDI Publications, 1968.
6. London, Keith R. Decision Tables. Princeton: Auerbach Publishers, 1972.
7. McDaniel, Herman. Decision Table Software. Princeton: Brandon/Systems Press, 1970.
8. _____. An Introduction to Decision Logic Tables. New York: Wiley, 1968.
9. _____, comp. Applications of Decision Tables. Princeton: Brandon/Systems Press, 1970.
10. Pollack, Solomon L.; Hicks, Harry T., Jr.; and Harrison, William J. Decision Tables: Theory and Practice. New York: Wiley-Interscience, 1971.